



Herramientas de configuracion

Gestión de Redes

- ❑ Los productos deben ofrecer **nuevas funcionalidades** para adaptarse a nuevas necesidades.
- ❑ Estas **funcionalidades** se seleccionan mediante **configuración**.

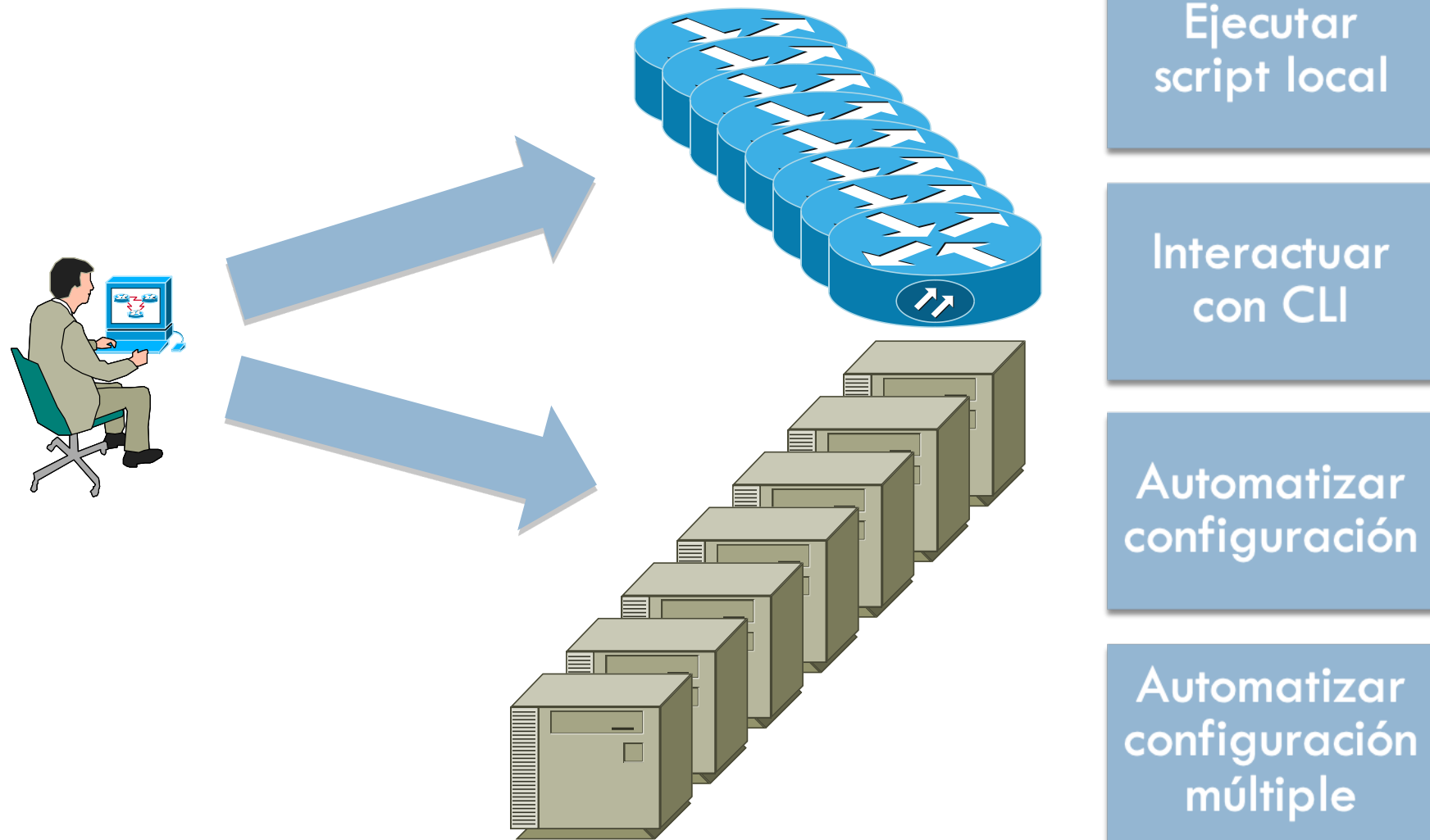
Configuración

- Valores de un conjunto de parámetros que un gestor debe elegir para controlar la operación de un dispositivo de red o servicio

- ❑ Los **parámetros** de **configuración** dan **flexibilidad** en el diseño de la red/servicio.
- ❑ Es complicado para un gestor hacer una configuración que prevea todas las circunstancias o **futuras necesidades**.
- ❑ Es necesario **reconfigurar** dispositivos/servicios.

¿Qué puedo hacer?

3



Lenguajes y herramientas

4

- **Lenguajes:** Shell, tcl, perl, php, rubi, VBScript, Jscript, python...
- **Generación de scripts:** Scriptomatic, WMI Code Creator, ADSI Scriptomatic (Active Directory), Tweakmatic, Log Parser, HTA Helpomatic, autoit....
- **Automatización procesos:** fileaze, FTP Scheduler, Workspace Macro....
- **Automatización de configuración:** expect, pexpect...
- **Sistemas avanzados de configuración automática:** puppet, BCFG, Chef, Cisco Intelligent Automation for Cloud, Ansible...



CFEngine

<http://cfengine.com>



<http://trac.mcs.anl.gov/projects/bcfg2/>



<https://puppetlabs.com>



<http://wiki.opscode.com>

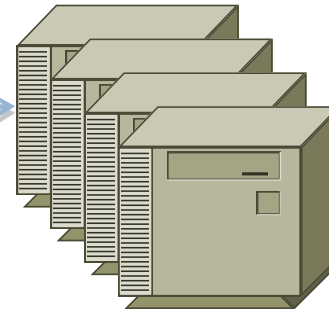
Expect

6

- ❑ Automatización CLI
- ❑ Conexiones telnet, ftp, ssh...



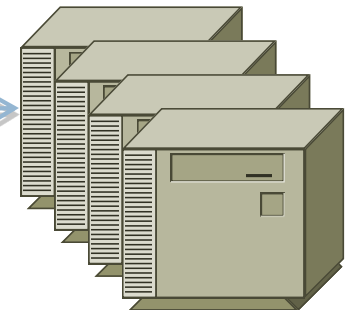
SSH



```
#!/usr/bin/expect -f
set password [lrange $argv 0 0]
set addr [lrange $argv 1 1]
set comm [lrange $argv 2 2]
set comm_arg [lrange $argv 3 3]
set timeout -1
spawn ssh root@$addr $comm $comm_arg
expect "*?assword:*"
send -- "$password\r"
send -- "\r"
expect eof
```

Expect
script

SSH



- **Expect:**
 - ▣ Lenguaje de Scripting: Extensión de TCL (Tool Command Language)
 - ▣ Automatizar interacción mediante CLI con aplicaciones (telnet, ftp, dispositivos...)
 - ▣ Soportado por varios sistemas operativos: Linux (original), Windows, OSX...
- **Pexpect:** módulo python expect

Características de Expect

8

- ❑ Envía comandos
- ❑ Espera respuesta a los comandos
- ❑ Soporta manejo de excepciones
- ❑ Interacciona con el usuario:
 - ▣ Pide datos
 - ▣ Informa resultado de un comando, error...
- ❑ Simula comportamiento humano en teclado
- ❑ Puede definir timeouts
- ❑ TCL:
 - ▣ Control de flujo: if, for, break...
 - ▣ Evaluación de expresiones, recursividad, definición de procedimientos...
 - ▣ Comandos TCL: if, exec, set...

- Instrucciones básicas:
 - ▣ **spawn**: conecta con sistema remoto (ftp, telnet, ssh...)
 - ▣ **send**: envía texto al proceso (conexión sistema remoto)
 - ▣ **expect**: espera respuesta
 - ▣ **send_user**: visualiza texto en la consola de usuario
 - ▣ **\n**: enter
 - ▣ *****: comodín
 - ▣ **exit**: terminar

Ejemplo de Script expect (I): login

10

```
#!/usr/bin/expect

set timeout 60

spawn ssh admin@10.0.0.1

send_user "intentando hacer login\r"
```

Ejemplo de Script expect (II): password

11

```
expect {  
    "Password:" send "1234abcd\r"  
    * {send_user "Error: no recibe petición de password\r"; exit}  
}  
  
sleep 2
```

Ejemplo de Script expect (III): configura interfaz

12

```
expect {  
    ">" {send "ifconfig eth0 up\r"; send_user "Activa interfaz eth0\r"}  
    *   {send_user "Error: no recibe el prompt\r"; exit}  
}
```

Ejemplo de Script expect (IV): fin

13

```
expect {  
    ">" send "logout\r"  
    * {send_user "Error: no recibe el prompt\r"; exit}  
}  
send_user "Termina\n"
```

- Expect es práctico en un sistema homogéneo:
 - ▣ Configurar un conjunto de equipos iguales
 - ▣ Con los mismos parámetros
 - ▣ Con las mismas credenciales
- Si el sistema no es homogéneo, la programación es más compleja

Expect con python: pexpect

15

- <http://pexpect.sourceforge.net/doc/>

```
# Conecta ftp server y descarga "reg0513.zip"
import pexpect
child = pexpect.spawn('ftp ftp.regserver.com')
child.expect('Name .*: ')
child.sendline('anonymous')
child.expect('Password:')
child.sendline('user@rs.com')
child.expect('ftp> ')
child.sendline('cd pub')
child.expect('ftp> ')
child.sendline('get reg0513.zip')
child.expect('ftp> ')
child.sendline('bye')
```

- ❑ Modelo cliente/servidor: Master/agent
- ❑ Basado en abstracción de recursos (RAL): file, package, user, service...
- ❑ Condicionales: if else, case...
- ❑ Declaración de variables: \$saludo = "Hola"
- ❑ Variables preasignadas (Facts): \$fqdn, \$ipaddress, \$hostname, \$operatingsystem

Puppet (cont.)

17

- **Agrupación** de recursos en una **clase** (class). Las clases pueden tener parámetros de entrada.
- **Reutilización** de código definido en **módulos**, guardados en un directorio específico con una estructura definida
<https://github.com/mv/puppet-sample/tree/master/puppet>
- Uso de **plantillas** con contenidos de configuración (template)

- ❑ Puppet usa abstracción de recursos (RAL).
- ❑ Un recurso puede ser: cuenta de usuario, fichero, proceso, software...
- ❑ Los recursos tienen asociados:
 - ❑ **tipo**
 - ❑ **nombre**
 - ❑ **atributos**
 - ❑ **valores**

User: root	User: oper1	User: oper2
• uid:0 • gid:0 • sh:/bin/sh	• uid:1 • gid:1 • sh:/bin/sh	• uid:2 • gid:1 • sh:/bin/sh

- Puppet usa RAL para gestión de configuración:
 - ▣ **Crear** nuevos recursos
 - ▣ **Leer** estado de recursos
 - ▣ **Verificar** si coincide con el estado deseado
 - ▣ **Modificar** estado si es necesario

Consulta estado del recurso root de tipo user

```
$ puppet resource user root
user { 'root':
  uid => 0
  gid => 0
  sh => '/bin/sh'
}
```

□ Tipos de recursos predefinidos:

- notify
- file
- package
- service
- exec
- cron
- user
- group
-

Obtener información sobre un tipo

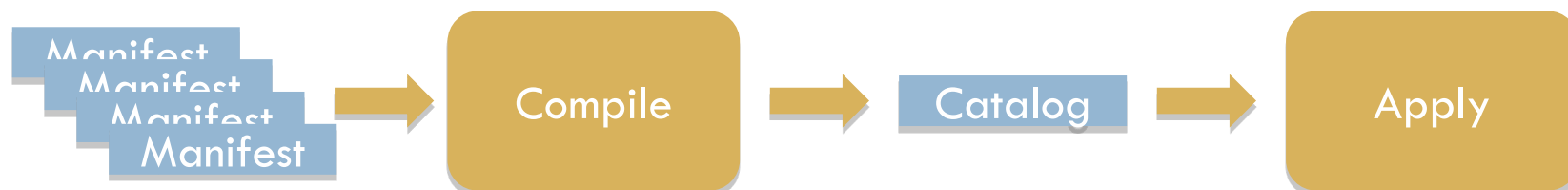
```
$ puppet describe --s user
```

Manifest & catalog

21

- Es un programa puppet con extensión .pp
- El **manifiesto** contiene información sobre configuración deseada de un recurso.
- Uno o más **manifiestos** se **compilan** en un **catálogo**.
- El **catálogo** representa recursos y el orden en que deben ser sincronizados.
- Un vez compilado el catálogo se procesa (**apply**).

```
$ puppet apply sample.pp
```



Ejemplo

22

ACTIVIDAD:

- ¿Qué recursos/s se configura/n?
- ¿Sobre que tipos de sistemas se aplica?
- ¿Sobre qué máquinas se aplica?
- ¿Qué acciones de configuración realiza?

exec: ejecuta comando

package: instala paquete

service: asegura que un servicio está corriendo

file: asegura que un fichero existe

```
#!/etc/puppetlabs/puppet/modules/ntp/manifests/init.pp
class ntp ($servers = undef, $enable = true, $ensure = running){
  case $operatingsystem {
    centos, redhat: {
      $service_name      = 'ntpd'
      $conf_template      = 'ntp.conf.el.erb'
      $default_servers    = [ "0.centos.pool.ntp.org",
                              "1.centos.pool.ntp.org",
                              "2.centos.pool.ntp.org", ]
    }
    debian, ubuntu: {
      $service_name      = 'ntp'
      $conf_template      = 'ntp.conf.debian.erb'
      $default_servers    = [ "0.debian.pool.ntp.org iburst",
                              "1.debian.pool.ntp.org iburst",
                              "2.debian.pool.ntp.org iburst",
                              "3.debian.pool.ntp.org iburst", ]
    }
  }

  if $servers == undef {
    $servers_real = $default_servers
  }
  else {
    $servers_real = $servers
  }

  package { 'ntp':
    ensure => installed,
  }

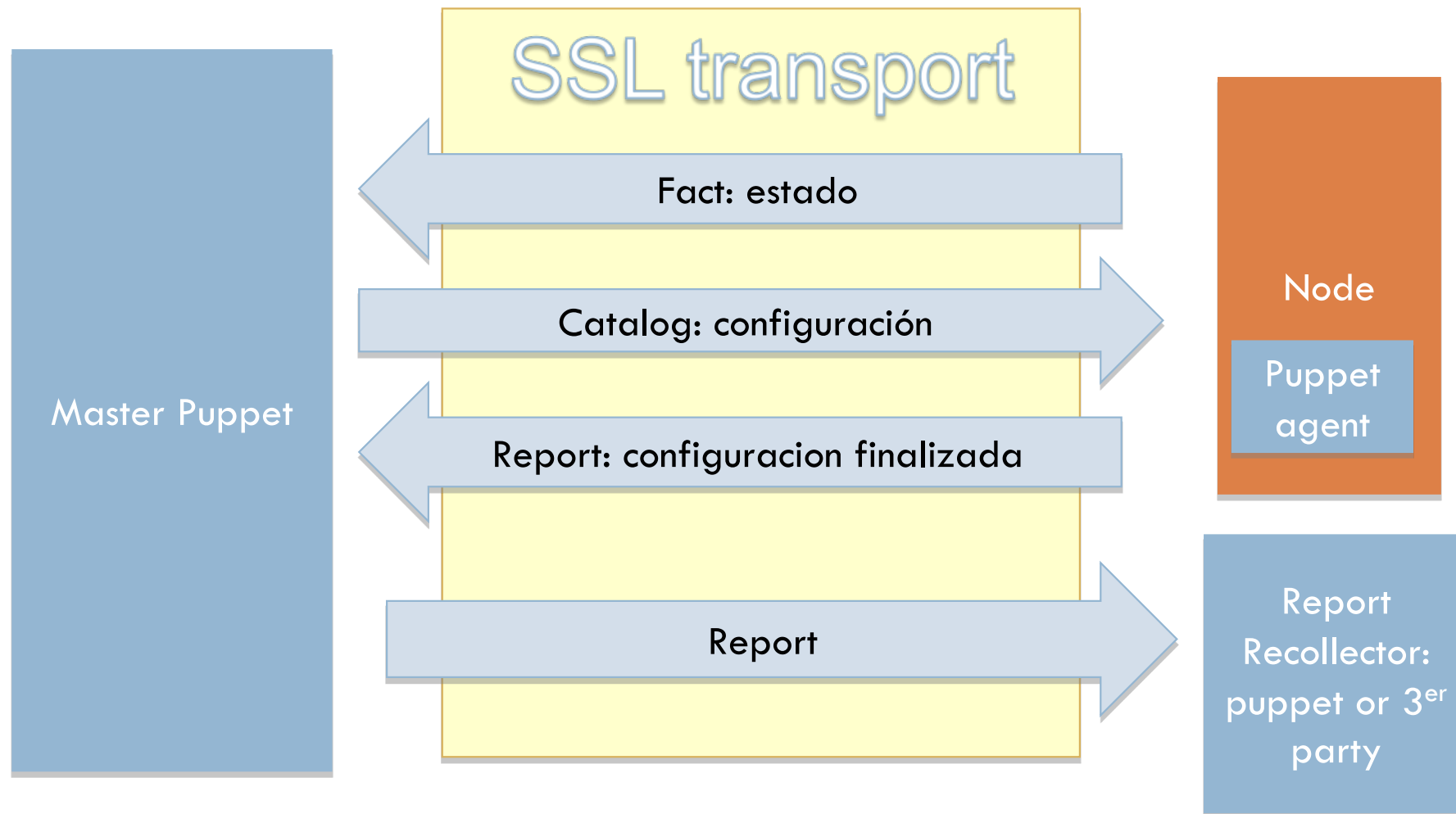
  service { 'ntp':
    name      => $service_name,
    ensure    => $ensure,
    enable    => $enable,
    subscribe => File['ntp.conf'],
  }

  file { 'ntp.conf':
    path      => '/etc/ntp.conf',
    ensure    => file,
    require   => Package['ntp'],
    content   => template("ntp/${conf_template}"),
  }
}
```

- **Define el estado deseado** de la configuración de la infraestructura usando el **lenguaje** declarativo de configuración **Puppet**.
- **Simula los cambios** de configuración antes de aplicarlos corrigiendo posibles errores.
- **Aplica** automáticamente la nueva configuración para pasar al estado deseado.
- Hace un **Informe** de la diferencia entre el estado actual y el deseado, y los cambios aplicados.

Arquitectura

24

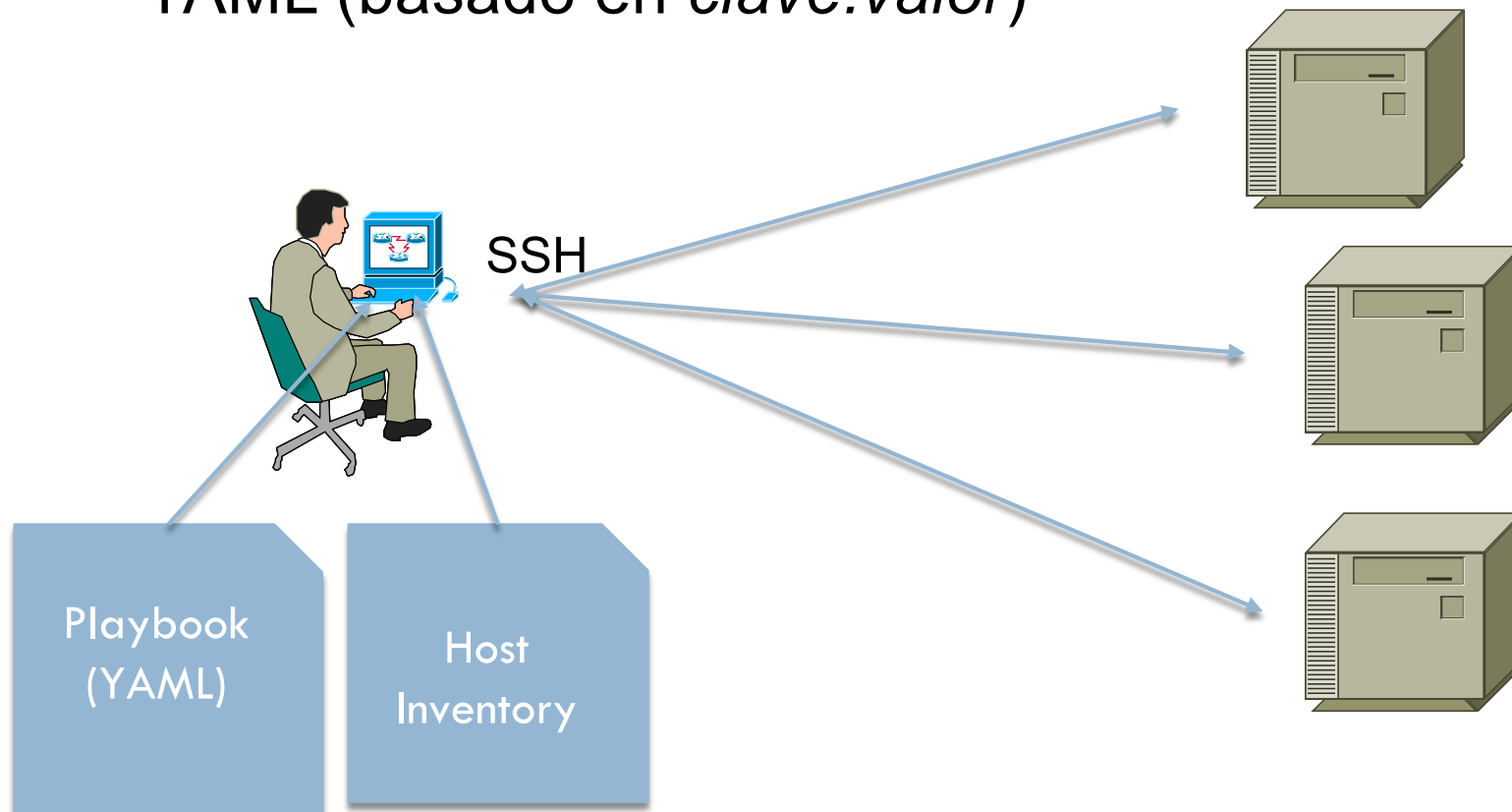


Ansible

- Herramienta de provisionado, para **instalar y configurar aplicaciones de forma centralizada y automática**
- Puede configurar sistemas, implementar software y organizar tareas avanzadas de IT como despliegues continuos o actualizaciones
- Objetivos de diseño: simplicidad, facilidad de uso, seguridad y fiabilidad

Ansible – ideas básicas

- Gestiona máquinas a través de conexiones automáticas usando SSH por defecto
- Define la configuración con el lenguaje de marcado YAML (basado en *clave:valor*)



Ansible-Inventario

- El inventario se encuentra en un fichero hosts y contiene los hosts que pueden ser configurados
- Se puede definir **grupos** de hosts
- Se pueden crear de forma dinámica

https://docs.ansible.com/ansible/latest/user_guide/intro_dynamic_inventory.html

Ejemplo:

www.uah.es

[webservers]

www1.uah.es

www2.uah.es

www3.uah.es

[dbservers]

db1.uah.es

db2.uah.es



Grupo



Grupo

Ansible-parámetros de inventario

Ejemplo de entrada en inventario con parámetros

[aut-node]

server1 ansible_host = <192.168.3.24> ansible_user = <admin>

ansible_pass = <hdX346%6hFFls>

□ Otros parámetros:

- ▣ Ansible_port: puerto de conexión, por defecto 22
- ▣ Ansible_connection: método de conexión, por defecto ssh

Ansible - Playbooks

- Las *playbooks* (libro de jugadas) son ficheros escritos en YAML que definen la forma en la que Ansible debe hacer la configuración para los diferentes equipos
- Cada *playbook* tiene una o más *plays* (jugadas)
 - ▣ Una *play* asigna a un grupo de hosts un conjunto de tareas (tasks) a realizar
 - ▣ Una tarea es una llamada a uno de los módulos de Ansible
 - ▣ Los módulos se envían al host para su ejecución y se borran
 - ▣ Se necesita el usuario remoto SSH para acceder al sistema y ejecutar la configuración

Ansible – Ejemplo de Playbooks

```
- hosts: snmp
  remote_user: admin
  become: yes
  become_method: sudo

tasks:
  - name: copy config file
    copy:
      src: /home/admin/snmpd.conf
      dest: /etc/snmp/snmpd.conf
      owner: root
      group: root
      mode: 0600

  - name: ensure snmpd is running
    service:
      name: snmpd
      state: started
```

SIGNIFICADO

Aplica sobre un grupo de host definido como snmp

El usuario de destino es admin (ssh a admin)

Hace operaciones sudo

Tareas:

Copia el fichero de configuración

Reinicia el servicio snmpd

¿COMO PROCESARLO?

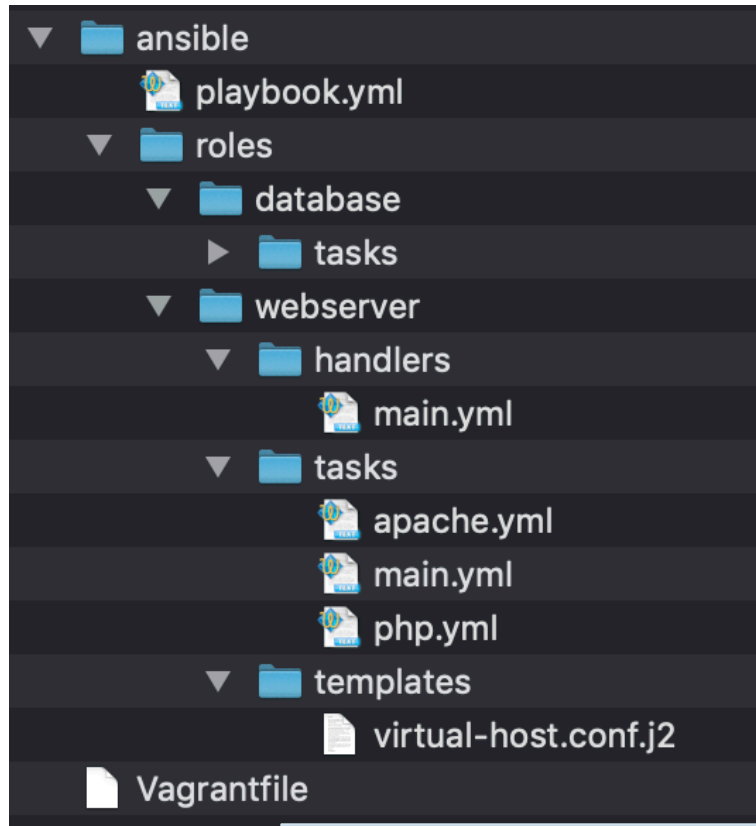
`ansible-playbook playbook.yml -f 5`

Caso de uso. Vagrant+Ansible

- Configurar LAMP:
 - ▣ Ubuntu server (VM)
 - Apache
 - Instalar
 - Configurar un virtual host llamado gr
 - PHP5: instalar
 - MySQL: instalar
 - ▣ Usar VirtualBox con vagrant+ansible

Estructura de ficheros...Vagrantfile

32



```
Vagrant.configure("2") do |config|  
  config.vm.box = "ubuntu/trusty64"  
  config.vm.network "forwarded_port", guest: 80, host: 8080  
  config.vm.provision :ansible do |ansible|  
    ansible.playbook = "ansible/playbook.yml"  
  end  
end
```

Vagrantfile

Vagrant en 30 segundos

33

□ Comando básicos

- ▣ Iniciar: `vagrant up`, Apagar: `vagrant halt`
- ▣ Reiniciar: `vagrant reload`,
- ▣ Suspende: `vagrant suspend` , Reanuda VM suspendida: `vagrant resume`
- ▣ Conectarse por ssh: `vagrant ssh`
- ▣ Ver el estado de vagrant: `vagrant global-status`, `vagrant status`
- ▣ Destruir: `vagrant destroy`
- ▣ Ver otra información: `vagrant port`, `vagrant version`
- ▣ Valida el fichero Vagrantfile: `vagrant validate`

□ Más avanzados

- ▣ Hacer snapshots de máquinas: `vagrant snapshot`
 - ▣ Opciones: `push`, `pop`, `save`, `restore`, `list`, `delete`
- ▣ Compartir maquina con otros: `vagrant share`

Volvemos a ansible

34

Configuración básica: playbook.yml

playbook.yml

```
- hosts: all
  become: true
  tasks:
    - name: update apt cache
      apt: update_cache=yes
    - name: install apache
      apt: name=apache2 state=present
    - name: install mysql
      apt: name=mysql-server state=present
    - name: install php
      apt: name=php5 state=present
```

Todas las máquinas

Privilegios de administrados (sudo)

Lista de tareas

name: nombre
asignado a la tarea
apt: módulo usado
para instalar
paquetes

Trabajando con variables

35

playbook.yml

```
---
- hosts: all
  become: true
  tasks:
    - name: update apt cache
      apt: update_cache=yes
    - name: install required packages
      apt: name= "{{packages}}" state=present
  vars:
    packages:
      - apache2
      - mysql-server
      - php5
```

Las variables facilitan las reutilización de Código

vars se puede poner en un fichero específico

Estructurando la información de configuración

36

- Defino dos **roles**:
 - ▣ Instalacion/configuración de web
 - ▣ Instalacion de base de datos
- Los puedo reutilizar en otras configuraciones
- **pre_task**: se ejecuta antes que roles. Su usamos **task** se ejecutaría después.

playbook.yml

- hosts: all
 become: true

pre_tasks:

- name: update apt cache
 apt: update_cache=yes

roles:

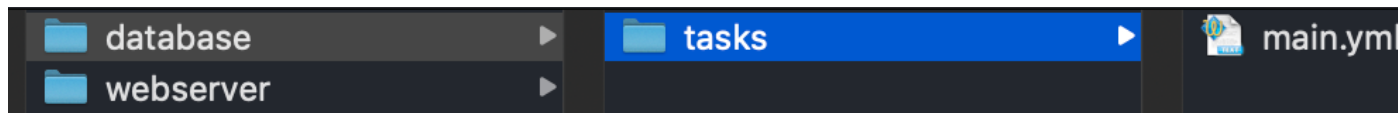
- webserver
- database



Instalación de base de datos

37

- ❑ Definición del rol de bases de datos.
- ❑ Instala la base de datos

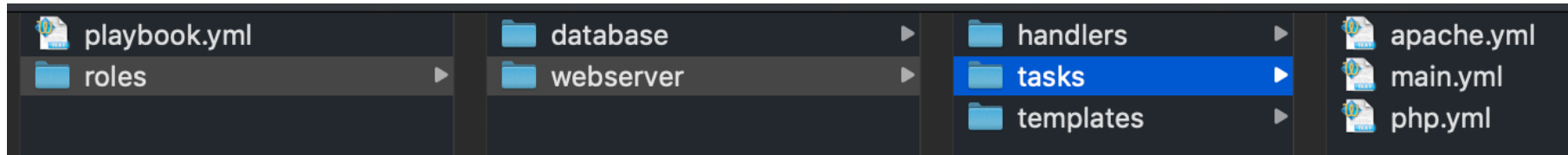


main.yml

```
- name: install mysql  
  apt: name=mysql-server state=present
```

Instalación/configuración de apache y php

38



main.yml

```
---  
- include: apache.yml  
- include: php.yml
```

Include, para hacer referencia a otros ficheros yml
Se ejecutan en el orden definido

php.yml

```
---  
- name: install php  
  apt: name=php5 state=present
```

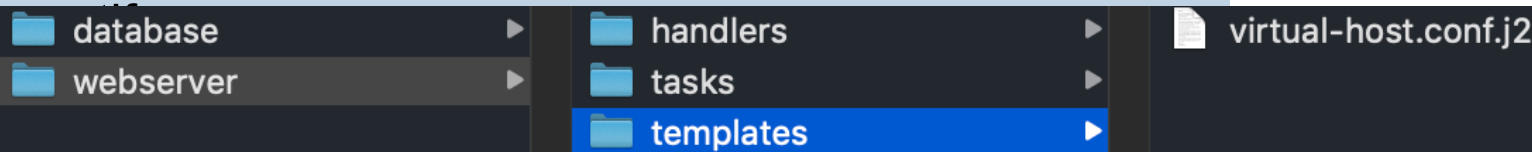
Instalación de PHP

Copiamos ficheros de configuración variables

39

apache.yml

```
---
- name: Install apache
  apt: name=apache2 state=present
- name: Copy across new virtual host
  template:
    src=virtual-host.conf.j2
    dest=/etc/apache2/sites-available/gr.conf
```



```
file:
  path=/etc/apache2/sites-enabled/000-default.conf
  state=absent
notify:
  - reload apache
- name: Enable new vagrant virtual host
  file: src=/etc/apache2/sites-available/gr.conf
      dest=/etc/apache2/sites-enabled/gr.conf
      state=link
notify:
  - reload apache
```

Con **template** copiamos un fichero local (de configuración) en VM. El template puede usar variables (**vars**)

En el ejemplo pasamos a apache el fichero de configuración para crear un nuevo virtual host llamado gr

La plantilla

40

```
<VirtualHost *:80>
    ServerAdmin webmaster@localhost
    DocumentRoot {{ document_root }}
    <Directory {{ document_root }}>
        Require all granted
    </Directory>
    ErrorLog ${APACHE_LOG_DIR}/error.log
    CustomLog ${APACHE_LOG_DIR}/access.log combined
</VirtualHost>
```

virtual-host.conf.j2

Definimos la variable en
playbook.yml, en este
ejemplo es el directorio
raíz para el web virtual
gr

playbook.yml

```
---
- hosts: all
  become: true
  vars:
    document_root: /gr
  pre_tasks:
    - name: update apt cache
      apt: update_cache=yes
  roles:
    - webserver
    - database
```

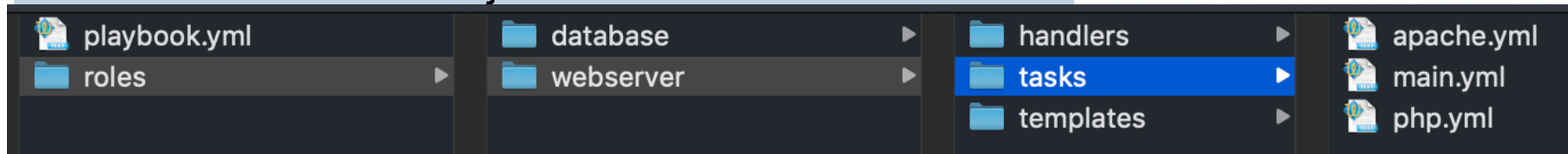
Borrando ficheros de configuración

41

apache.yml

```
---  
- name: Install apache  
  apt: name=apache2 state=present  
- name: Copy across new virtual host  
  template:  
    src=virtual-host.conf.j2
```

Con el modulo **file**
Podemos borrar un
fichero (state = absent)



```
- name: Remove default virtual host  
  file:  
    path=/etc/apache2/sites-enabled/000-default.conf  
    state=absent  
  notify:  
    - reload apache  
- name: Enable new vagrant virtual host  
  file: src=/etc/apache2/sites-available/gr.conf  
        dest=/etc/apache2/sites-enabled/gr.conf  
        state=link  
  notify:  
    - reload apache
```

Borramos el fichero de
configuración, por
defecto de virtual host
en la configuración de
apache

Links a ficheros: symlink

42

apache.yml

```
---
- name: Install apache
  apt: name=apache2 state=present
- name: Copy across new virtual host
  template:
    src=virtual-host.conf.j2
    dest=/etc/apache2/sites-available/gr.conf
  notify:
    - reload apache
- name: Remove default virtual host
  file:
    path=/etc/apache2/sites-enabled/000-default.conf
    state=absent
  notify:
    - reload apache
- name: Enable new vagrant virtual host
  file: src=/etc/apache2/sites-available/gr.conf
        dest=/etc/apache2/sites-enabled/gr.conf
        state=link
  notify:
    - reload apache
```

Con el modulo **file**
Podemos hacer enlaces
simbólicos de ficheros
(state = link)

Creamos un link del
fichero de configuración
del host virtual gr para
que esté presente en la
carpeta sites-enabled
(activo)

!Manejándo manejadores!...handler

43

apache.yml

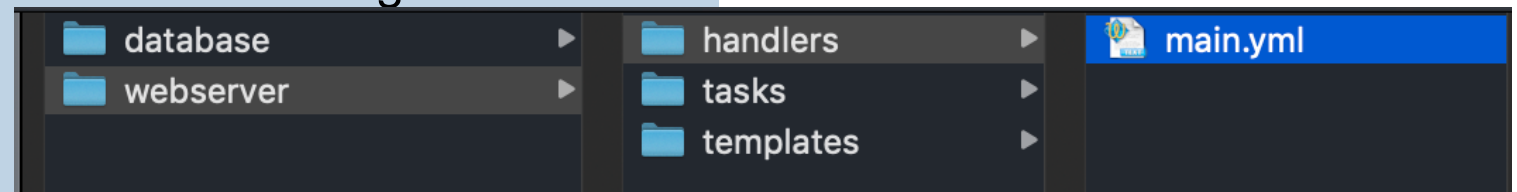
```
---
- name: Install apache
  apt: name=apache2 state=present
- name: Copy across new virtual host
  template:
    src=virtual-host.conf.j2
    dest=/etc/apache2/sites-available/gr.conf
  notify:
    - reload apache
- name: Remove default virtual host
  file:
    path=/etc/apache2/sites-enabled/000-default.conf
    state=absent
  notify:
    - reload apache
- name: Enable new vagrant virtual host
  file: src=/etc/apache2/sites-available/gr.conf
        dest=/etc/apache2/sites-enabled/gr.conf
        state=link
  notify:
    - reload apache
```

Con **handler** realizamos una acción cuando la llamamos con **notify**

handlers/main.yml

```
---
- name: reload apache
  service: name=apache2 state=reloaded
```

En este ejemplo reinicamos apache después de cada cambio de configuración



- Automated Network Management Systems, 2006. ISBN 0-13-239308-5. Douglas Comer.
- Expect:
<http://expect.nist.gov/>
<http://expect.sourceforge.net>
<http://www.tcl.tk/man/expect5.31/expect.1.html>
- Ejemplos de gestión con expect
<http://www.tcl.tk/customers/success/netmanage.html>
- Módulo python de expect: pexpect
<http://www.noah.org/wiki/Pexpect>
- Puppet
<https://puppetlabs.com/>
- Chef
<http://www.opscode.com/chef/>
- Ansible
<http://docs.ansible.com/>
<https://www.tutorialspoint.com/ansible/index.htm>
<https://adamcod.es/2014/09/23/vagrant-ansible-quickstart-tutorial.html>